Optimized Task Scheduling for Cloud Computing Using PSO and GA

B.V.seshu Kumari, Giri Prasad, Somula Ramasubbareddy, K.Govinda

Abstract*: Cloud computing started with risk-free concept: Let someone else take the ownership of setting up of IT infrastructure and let end-users tap into it, paying only for what is been used. A service offering computation resources is frequently referred to as Infrastructure as a Service (IaaS) and Software as a Service (SaaS). An environment used for construction, deployment, and management of applications is called PaaS (Platform as a Service). With the advent of cloud computing, resizable infrastructure for data analysis is now available to everyone via an on-demand maybe free model. Cloud computing totally depends on the internet to deliver its services to the users. In the modern computing environment where the amount of data to be processed is increasing day by day, the costs involved in the transmission and execution of such amount of data is mounting significantly. So there is a requirement of appropriate scheduling of tasks which will help to manage the escalating costs of data intensive applications. In this paper, we propose an algorithm that uses particle swarm optimization to schedule the tasks to get maximum benefits from the resources. The benefit can be expressed in terms of increased resource usage, minimized response time, minimize overall cost incurred. We will then schedule the tasks using genetic algorithm and compare the results given by particle swarm optimization as to what extent they are similar and which gives better results. Both algorithms are heuristic algorithms hence result may vary.

Keywords:Cloud Computing, Particle Swarm Optimization, Genetic Algorithm, Objective function..

I. INTRODUCTION

Among many evolving technologies, cloud computing is expanding its utilization in our day to day life. The popular technology, gained demand in the market because of its budget friendly nature. Generally, the software and hardware system costs high to acquire their use. This has been the disadvantage for getting appropriate for many organizations and corporations. To get the optimal solution to overcome the disadvantage using cloud computing and came out with solution providing the software and hardware for low price compared to the other. Unlike other sources for users to purchase the software and hardware systems, they are provided with resources easily as electricity is available commonly to everyone. Just as the other resources we get for our home, cloud computing is easily available to users without bothering about the source of the resources they are coming from. Some research areas like Grid computing, distributed computing and virtualization etc. are evolved from cloud computing as their origin. Cloud computing is most selective for another reason that is because of its characteristics. Cloud computing became an eminent technology due to loose coupling, service destined, resource pooling. If the user have internet with him, the availability of cloud computing is very easy.Resource allocation may be a difficult issue as a result of the planning of resources becomes tougher once the range of users becomes large enough and asks for identical resources at the same time. In general, task scheduling makes a single task into subtasks and assign them to various servers to the execution of task more efficiently with less utilization of resources such as memory and time. But task scheduling is used for a different purpose in cloud computing to maximize the utilization of cloud computing and profit. Of course it is not that easy to achieve high profit and utilization. Here with some constraints the process of execution of tasks is a key purpose of scheduling whereas in general allocation of resources have much importance to get optimized utilization of execution time and Qos

Revised Manuscript Received on July 05, 2019.

B.V.seshu Kumari, Associate Professor, VNRVJIET, Hyderabad,
Giri Prasad .A, Assistant Professor, VNRVJIET, Hyderabad, Telangana
Somula Ramasubbareddy, VNRVJIET, Hyderabad, Telangana, Asst.Professor
K.Govinda, VIT University, Vellore, Tamilnadu, Associate Professor
Mail: svramasubbareddy1219@gmail.com.

2 Background.

Task scheduling is important for both cloud as well as grid computing. Scheduling plays an important role in optimized utilization of resources which in turn leads to maximum benefit from resources. Many scheduling algorithms have been designed for optimizing the utilization of resources. This also includes heuristic algorithms like particle swarm optimization and genetic algorithm. Disk space management has been calculated as a critical issue in virtual environment after analyzing many task scheduling algorithm and their issues [1]. According to this paper Batch Mode Heuristic Scheduling Algorithms (BMHA) and Online Heuristic Scheduling Algorithms lack in reliability and availability although throughput produced by them is high and cost effective [2]. By comparing Particle Swarm Optimization with best resource selection yields a result that PSO performs well in reducing the cost of execution and also helps well in distributing resources [3][4]. Particle Swarm Optimization include a theory called Pareto dominate theory which is introduced to deal with the resource allocation [5]. Pareto dominate theory works to find out optimal scheduling considering total task execution time, resource allocation and Qos of each task [5]. Inter cloud resources are scheduled effectively by utilizing a self –adaptive learning (SLPSO) approach in which Iaas provides its work load to other

external clouds in case of resources deficiency. Author in [7] came out with a result after comparing SLPSO with cloud federation as the approach that best suits for finding optimal and sub-optimal allocation of internal and external resources and also utilized to maximize the quality and the profit of Iaas provider in SLPSO.

a. Practical Swarm Optimization

Kennedy and Eberhart in [8], introduced a technique similar to Genetic Algorithm called Practical Swarm Optimization. However, PSO is similar to Genetic Algorithm (GA), it is slightly different in recombining of individual of the population. In GA, recombination of individual is direct but not in PSO. Since it lacks in convergence accuracy it became popular for many other features like less computational costs and high speed. The solution evolved as a cluster of particles in which each particle consists of velocity and position. At first, particles and velocity vectors are initialized to random positions and search space respectively. After initializing velocity and position calculate fitness value for each of the particle. Now compare fitness value and pbest value, pbest=p if fitness value of p is greater than pbest. Among all the particles store the best particle in gbest. Finally update the values of each particle with help of equations (1) and (2).

$$V_i^{t+1} = wV_i^t + C_1 r_1 (pbest_i - X_i^t) + C_2 r_2 (gbest_d - X_i^t) \quad (1)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad (2)$$

V_i^t = Velocity of i th particle at iteration t

V_i^{t+1} = velocity of i th particle at iteration t + 1

$pbest_i$ = Bestpositionofithparticle

$gbest_d$ = Positionofbestparticleinapopulation

X_i^t = Currentpositionofithparticleatiterationt

X_i^{t+1} = Positionofithparticleatiterationt + 1

$C_1 C_2$ = AcceraltionConstants

r_1 and r_2 = RandomValues

W= Inertia weight

b. Algorithm 1 – Particle Swarm Optimization (PSO) algorithm

Step 1. Initialize particles with some random position and velocity vectors.
Step A. Calculate fitness value for every particle.
Step B. If fitness(p) better than fitness(pbest) than pbest=p.
Step C. Assign best particle pbest value to gbest.
Step D. Calculate velocity for each particle using equation (1).
Step E. Update particle velocity and position using equation (2).

Inertia weight helps in increasing overall performance of PSO. Global exploration and local exploration are increased based on the inertia weight. If the weight is small, local exploration increases and when weight is more, global exploration goes high. Appropriate selection of inertia leads to balance the global and local exploration. Number of loops also can be brought down by using inertia weight that helps in decreasing time and memory consumption. Performance is inversely proportional to inertia weight. In past, PSO solved many combinational optimization problems. Applications like Chemical Engineering [9], pattern recognition [10] and reactive voltage control problem come under PSO applications.

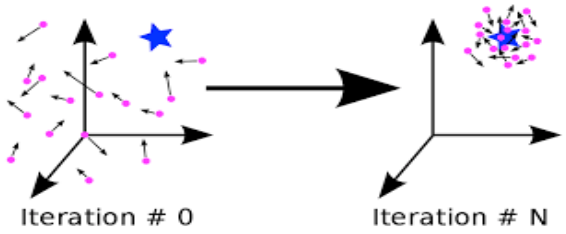


Fig. 1. Functioning of Practical Swarm Optimization

4 Genetic Algorithm

A process in evolutionary algorithms called natural selection inspired a genetic algorithm in computer science. High quality solutions are obtained using this meta- heuristic (GA). Mutation, cross over and selection are the operators on which Genetic Algorithm searches problems. The properties of found solutions such as chromosomes or genotype are represented in binary values in the combination of 0's and 1's and these can be mutated and altered.

The evolution is iterative process. A group of individuals are formed as population. The formed population in each iteration known as generation. Generation process is used to calculate the fitness of each and every single individual who are formed as population. The optimization problem is to be solved maintains fitness as its object value. New generations are formed by each individuals genome which are picked from the current population as they are more fit. The new generations formed by this process will be used in further iterations. This iteration continues till the count reaches maximum no. of

generations (or) when the fitness level shows the satisfying result

A typical genetic algorithm requires:
The requirements that are to be performed at the initial stage is

a genetic representation of the solution domain.
a fitness function to evaluate the solution domain.

After defining the requirement of genetic algorithm i.e. genetic representation and fitness function. Population of solution are initialized and are improved by repeating application of bio-inspired operators such as mutation, cross over, inversion and selection.

4.1 Initialization

Based on the nature of the problem, size of population varies. Although, the population have hundreds or thousands of possibilities in obtaining solutions. However all the solutions cannot not used at the same time. At initial stage, population allows all the range of possibilities of solutions. According to the requirement solutions are used only when optimal solutions are needed to be found.

4.2 Selection

A new generation is formed from the part of the existing population in every successive generation. The process to select the solution is based on the fitness. However, only the individual solutions are selected. This type of selection leads to best solutions and to rate every individual fitness.

4.3 Genetic operators

The next step is to generate a second generation population of solutions from those selected through a combination of genetic operators: crossover (also called recombination), and mutation. For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. When only the best organisms are selected from first generation and only some amount of less fit solutions for breeding can lead to increase average fitness.

4.4 Heuristics

The heuristic operators that have been discussed in the above section are not enough to increase the speed of calculation or robustness. Rather some more heuristics are needed to be added to increase the performance.

4.5 Termination

The generational process gets terminated when these scenarios are occurred

a) A satisfying solution is obtained
b) Generation process reaches the end
c) Resources like time/memory reaches the end of the given range.
d) No further iterations give optimal solution.

Manual inspection

Combinations of the above

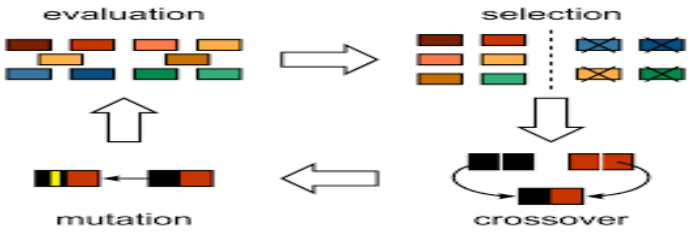


Fig. 2. Functioning of Genetic Algorithm

5 Proposed Work

First, we generate the problem matrix which represents the various tasks and their attributes which need to be scheduled in the cloud. It is a n x m x 3 matrix where:
n: no of tasks that need to be scheduled
m: no of dedicated machines to which the tasks can be assigned
Let P be the problem matrix.
The third dimension represents the following information:
P[i][j][0]: Time elapsed (in hours) in executing the i^{th}job on the j^{th} machine

P[i][j][1]: The cost (in dollars) of executing the i^{th}job on the j^{th} machine

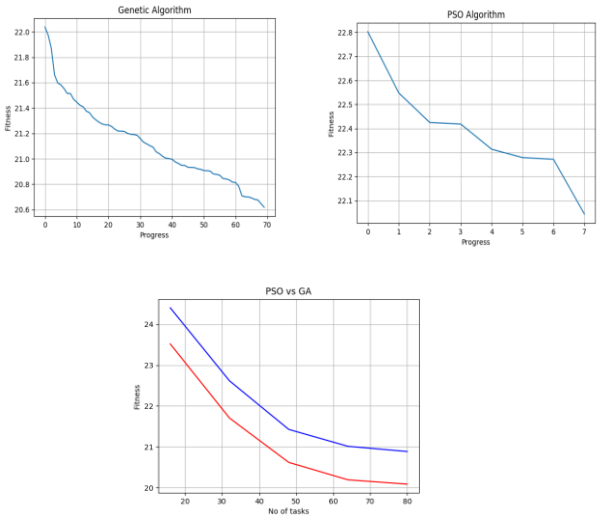
P[i][j][2]: The no of nodes available if i^{th}job is assigned to the j^{th} machine

5.1 Objective function

fitness = wt1* normalized_cost + wt2 * normalized_time + (1-wt1-wt2) * normalized_nodes
where ,
normalized_cost = cost incured / total cost sum of the entire matrix
normalized_time = time incured/total time sum of the entire matrix
normalized_nodes = (nd-total no of nodes taking) /nd
nd : sum of the total no. of nodes of the entire matrix
w1 and w2 being the tuning factors to manipulate the weight age of each of the three parts of
The fitness functions as per requirements.
The given tasks should be assigned to the various machines in such a way that the value of the fitness function is minimized.

5.2 Methodology

First, we use particle swarm optimization for this task scheduling problem. The no of particles defined and the number of iterations depends on the number of jobs. More specifically,
no of particles = 4 * number of jobs
no of iterations = 3 * number of jobs
Each particle is an array of size equal to the number of jobs. Each job is randomly assigned to one of the dedicated machines. Then we calculate the fitness function associated with each particle and follow the particle swarm optimization algorithm such that each particle tries to replicate the best particle with an aim to minimize the value of the fitness function. Finally, the global best variable holds the best possible solution as per the pso algorithm. This result is stored. The i^{th} position of the array stores the index of the machine that the i^{th} job needs to be assigned to. Now, we use genetic algorithm for the same problem. The size of the population depends on the no. of jobs. More specifically,
Population size = 4 * number of jobs
Each member of the population is of size equal to the number of jobs. Each job is randomly assigned one of the dedicated machines. Then we calculate the fitness function associated with each member and follow the genetic algorithm and breed members accordingly. The best solution achieved so far is stored. The i^{th} position of the array stores the index of the machine that the i^{th} job needs to be assigned to. Now we plot graphs and compare the results obtained from the two algorithms.



Blue: PSO
Red: GA

Fig. 3. Comparison of PSO and GA

6 Conclusion

The number of jobs and machines are changed for every execution of the algorithm. Though the fitness values produced by both PSO and GA were comparable, GA always outperformed PSO. Thus GA gives better (lower) fitness value than PSO. This conclusion is also evident from the above graph. Thus task scheduling by GA will produce more benefits than that by PSO. There is direct affect in the performance of cloud computing due to the scheduling algorithms efficiency. PSO convergence rate will be row in solving large scale of optimization problems. It is good in the initial stage but with each iteration particles begin to lose variety.

References

1. [1] P. Salot, "A survey of various scheduling algorithm in cloud computing environment," *Int. J. Res. Eng. Technol.*, vol. 2, no. 2, pp. 131–135, 2013.
2. [2] R. Kaur and P. Luthra, "Load balancing in cloud computing," in *Proceedings of International Conference on Recent Trends in Information, Telecommunication and Computing, ITC*, 2012.
3. [3] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *Advanced information networking and applications (AINA), 2010 24th IEEE international conference on*, 2010, pp. 400–407.
4. [4] H. Zhang, P. Li, Z. Zhou, and X. Yu, "A PSO-based hierarchical resource scheduling strategy on cloud computing," in *International Conference on Trustworthy Computing and Services*, 2012, pp. 325–332.
5. [5] M. Feng, X. Wang, Y. Zhang, and J. Li, "Multi-objective particle swarm optimization for resource allocation in cloud computing," in *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, 2012, vol. 3, pp. 1161–1165.
6. [6] X. Zuo, "Guoxiang zhang," *Wei Tan,—Self-Adaptive Learn. PSO-Based Deadline Constrained Task Sched. Hybrid IaaS Cloud‖ IEEE Trans. Autom. Sci. Eng.*, 2013.
7. [7] R. Akbari and K. Ziarati, "Combination of particle swarm optimization and stochastic local search for multimodal function optimization," in *Computational Intelligence and Industrial Application, 2008. PACIIA'08. Pacific-Asia Workshop on*, 2008, vol. 2, pp. 388–392.
8. [8] R. Kennedy, "J. and Eberhart, Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks IV, pages*, 1995, vol. 1000.
9. [9] H. Yu, L. Zhang, D. Chen, X. Song, and S. Hu, "Estimation of model parameters using composite particle swarm optimization," *J. Chem. Eng. Chinese Univ.*, vol. 19, no. 5, p. 675, 2005.
10. [10] M. G. H. Omran, "Particle swarm optimization methods for pattern recognition and image processing," 2006.
11. [11] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Trans. power Syst.*, vol. 15, no. 4, pp. 1232–1239, 2000.